Architecture Overview Diagram & Component Model

An introduction to these key work products
Learning Objectives

At the end of this lecture, you should be able to:

- Understand:
  - What is an Architecture Overview Diagram (AOD)
  - What uses are there for an Architecture Overview Diagram
  - What is a Component Model and how is it represented
  - How an AOD and a Component Model relate to an Operational Models

- Develop a simple Architecture Overview Diagram
- Identify potential issues when reviewing an Architecture Overview Diagram
- Identify candidate components for a Component Model
Architecture Overview Diagram

What is it?
Where does it fit?
Examples
What is an Architecture Overview Diagram?

The purpose of this work product is:

- To **communicate** to the **sponsor** and external stakeholders a conceptual understanding of the intended IT system.
- To provide a **high-level shared vision** of the architecture and scope of the proposed IT system for the development teams.
- To explore and **evaluate alternative architectural options**.
- To enable early recognition and validation of the **implications of the architectural approach**.
- To facilitate **effective communication** between different communities of stakeholders and developers.
- To facilitate orientation for new people who join the project.

**Important things to note:**

- An Architecture Overview Diagram contains schematic diagrams that represent the governing ideas and building blocks of an IT system.
- An AOD can include both functional and operational concepts.
- An AOD is not a model.
Where does the Architecture Overview Diagram fit?

Requirements

- Use cases
- NFRs
- System context
- Existing IT
- ... and so on

Architecture overview diagram

"in the middle!"

IT Solution

Component model

Operational model
Example 1: Retail multi-channel access

The Retail Customer can choose from a variety of ways to interact with the company. The supporting infrastructure should be common whenever possible.
Example 2: Corporate applications

- **Off line Apps**
- **Dial-in**
- **BE Intranet**
  - eMail
  - ICM
  - etc
  - Business Services
- **FE Intranet**
- **Local Applications**
  - "off line" fat client
  - "on line" fat client
  - "on line" thin client
- **General Services**
  - Access Control
  - File Services
  - Service Management
Example 3: Local e-government

Public access

Mediated access

Partners, Agencies

Firewall layer

HTML/HTTP

HTML/HTTP

HTML/HTTP

XML/SOAP/HTTP

XML/SOAP/etc
eGIF standards

Transformation & Presentation Layer

Personalization, Transaction definition & recording

CRM system

Transactions

Workflow, application routing

Web services layer

Services

Existing web servers

Potential ad hoc direct VPN connections

Allow for perpetual variety

Back end servers
Example 4: e-business Reference Architecture

<table>
<thead>
<tr>
<th>Users</th>
<th>Delivery Channels</th>
<th>e-business Services</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Pervasive Wireless Devices</td>
<td>Registration Function</td>
<td>Directory Systems</td>
</tr>
<tr>
<td>Service</td>
<td>Internet Browser</td>
<td>Authentication and Authorization Function</td>
<td>Legacy Applications</td>
</tr>
<tr>
<td>Representative</td>
<td>Intranet Browser</td>
<td>Enterprise Inquiry Function</td>
<td>Database(s)</td>
</tr>
<tr>
<td>Internal User</td>
<td>Internet or Extranet Browser</td>
<td>Enterprise Update Function</td>
<td>System Monitoring</td>
</tr>
<tr>
<td>Business Partner</td>
<td></td>
<td>Enterprise Reporting Function</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Messaging &amp; Collaboration Function</td>
<td>External Enterprise System</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enterprise Administration Function</td>
<td></td>
</tr>
</tbody>
</table>
Component Model

What is a Component?
What is a Component Model?
How do you create one?
The primary concept used for modular design

Within the software domain, a component can be defined as “…an encapsulated part of a software system that provides a well-defined interface to its services”

Components are not limited to application components. They can also be:

- Technical components
- System software components
- Hardware components

examples?
Components are a formal modelling construct

- Components can be comprised of other components

- A subsystem groups components, but cannot be characterized as a component because it does not have interfaces.

- Objects are not very good or useful components

*Why?*
The notation used to represent components is based on UML

- Component representation uses UML Class notation

- Component interfaces specify their services
The function of an IT System is described by components

- **Components**
  - Are identified based on their responsibilities that collectively achieve the system behavior

- **Component Interfaces**
  - Represent an agreement of the requested services that describes component responsibilities and access to the interfaces’ data

- A component is **developed through several stages**, including:
  - Component **identification**
  - Component **specification**
  - Component **realisation**
Component Models include two types of diagram

- **Component Relationship Diagram** (Static Model)
  - Is represented by a variation of the UML Class Diagram

- **Component Interaction Diagram** (Dynamic Model)
  - Depicts component relationships and dependencies
  - Illustrates how components collaborate to achieve system functionality
  - Is represented by a variation of the UML Collaboration or Sequence Diagram

A Component model is *never just one diagram*
A Component Model is used to describe complex software solutions

- A Component Model helps to bridge the gap between requirements and the solution by:
  - Ensuring that detailed specifications need not be made immediately but can be elaborated over a period of time
  - Mandating the main design principles and overall structure

- The Component Model achieves this by defining smaller problem scopes that can be handed to different teams while encouraging reuse.

- Each of these problem scopes can then have an associated:
  - Analysis and detailed design
  - Implementation
  - Logical and physical database model
Component modeling is divided into three stages

- High level design focuses on component **identification**
- Detailed design deals with component **specification**
- Development deals with component **realisation**

Component Identification
- Generate Candidate Components
- Structuring
- Partitioning
- Layering

Component Specification
- Define Component Interfaces
- Assign Business Rules
- Specify Components

Component Realisation
- Perform Product Selection
- Determine Implementation Approach
The Architecture Overview Diagram of a Home Shopping Example

Users
- Corporate User (Online)
- Private User (Online)
- Head Office IRP
- Country IRP
- In-store IRP

Delivery Channels
- Internet Browser
- Intranet Browser

Business Services
- Registration and Profile Management
- Browse and Search Services
- E-commerce Services
- Download Services
- Store Services
- Jobs Services
- Press Services
- Customer Services
- Site Statistics Services
- Content Management
- Site Administration

Technical Services
- Integration Hub
- Legacy Systems
- Other Services
The Component Relationship Diagram shows the static relationships among components.
Components are identified, named and their responsibilities are described

- <COMP-001> ProductMgr

- The product manager component is responsible for interacting with back-end systems and providing product, article, and category information. Conceptually, the component performs a batch job at a set schedule, performing the following actions:
  - Querying back-end systems for new or updated products/articles (items)
  - Extracting information from the back-end system
  - Possibly transforming or filtering the information
  - Responding to real-time queries to provide product information
Component Interaction Diagrams show the dynamic relationships among components

Also known as sequence diagrams.
The Component Collaboration Diagram is a different way of looking at the Dynamic Model

1: request product page
2: get product page
3: change options
4: add to list
5: add item
Architecture Overview Diagram & Component Model

Summary
Learning Points

- Use an Architecture Overview Diagram to provide effective communication between different communities of stakeholders and developers.

- An Architecture Overview Diagram is not a model.

- Components are the software building-blocks of an IT system, providing services through their interfaces.

- Component Models describe the static relationships and the dynamic interactions between components.